NASA/CR—1998-207401

EUPDF - An Eulerian-Based Monte Carlo Probability Density Function (PDF) Solver

User's Manual

M.S. Raju
NYMA, Inc., Brook Park, Ohio

Prepared under Contract NAS3–98022

National Aeronautics and
Space Administration

Lewis Research Center

April 1998

EUPDF - an Eulerian-Based Monte Carlo Probability Density Function (PDF) Solver - User's Manual

M.S. Raju
Nyma, Inc., NASA Lewis Research Center
2001 Aerospace Parkway
Brook Park, Ohio-44142

Abstract

EUPDF is an Eulerian-based Monte Carlo PDF solver developed for application with sprays, combustion, parallel computing and unstructured grids. It is designed to be massively parallel and could easily be coupled with any existing gas-phase flow and spray solvers. The solver accommodates the use of an unstructured mesh with mixed elements of either triangular, quadrilateral, and/or tetrahedral type. The manual provides the user with the coding required to couple the PDF code to any given flow code and a basic understanding of the EUPDF code structure as well as the models involved in the PDF formulation. The source code of EUPDF will be available with the release of the National Combustion Code (NCC) as a complete package.

Table of Contents

I. Nomenclature

A	pre-exponential coefficient in an Arrhenius reaction-rate term
$\underline{a}_n$	outward area normal vector of the nth surface, m^2
C_p	specific heat, J/(Kg K)
c_n	convection/diffusion coefficient of the nth face, kg/s
D	turbulent diffusion coefficient, m^2/s
E_a	activation energy in an Arrhenius reaction-rate term
h	specific enthalpy, J/Kg
J_i^α	diffusive mass flux vector, Kg/ms
k	turbulence kinetic energy, m^2/s^2
M_i	molecular weight of ith species, kg/kg-mole
m_k	droplet vaporization rate, Kg/s
$l_{k,eff}$	effective latent heat of evaporation, J/Kg
N_{av}	number of time steps employed in the PDF time-averaging scheme
N_f	number of surfaces contained in a given computational cell
N_m	total number of Monte Carlo particles per grid cell
N_p	total number of computational cells
n_k	number of droplets in kth group
P	pressure, N/m^2
P_r	Prandtl number
p	joint scalar PDF
R_u	gas constant, J/(Kg K)
r	radial coordinate (gas-phase equations), m
S_α	liquid source contribution of the α variable
S_c	Schmidt number
S_{mlc}	liquid source contribution of the gas-phase continuity equation
S_{mle}	liquid source contribution of the gas-phase energy equation
S_{mlm}	liquid source contribution of the gas-phase momentum equations
S_{mls}	liquid source contribution of the gas-phase species equations
T	temperature, K
t	time, s
u_i	velocity component in the ith direction, m/s
w_α	chemical reaction rate, 1/s
x_i	Cartesian coordinate in the ith direction, m
Y_j	mass fraction of jth species
$\underline{x}$	spatial vector

greek symbols

Γ_ϕ	turbulent diffusion coefficient, kg/ms
Δt	local time step used in the PDF computations, s
Δt_f	local time step in the flow solver, s
ΔV	computational cell volume, m^3
δ	Dirac-delta function
$\epsilon_{\alpha s}$	species mass fraction at the droplet surface
ϵ	rate of turbulence dissipation, m^2/s^3
λ	thermal conductivity, J/(ms K)
μ	dynamic viscosity, kg/ms
ρ	density, kg/m^3
χ	mole fraction
σ	dimensionality of $\underline{\psi}$-space
$\underline{\psi}$	independent composition space
ω	turbulence frequency, 1/s
τ	stress tensor

superscripts

$\tilde{}$	Favre averaging
$\bar{}$	time averaging or average based on the Monte Carlo particles present in a given cell
$''$	fluctuations

subscripts

α	scalar variable of the joint PDF
f	represents conditions associated with fuel
g	global or gas-phase
i	coordinate or specie indices
j	specie indices
k	droplet group or liquid phase
l	liquid phase or laminar
m	conditions associated with N_m
n	nth-face of the computational cell
o	initial conditions or oxidizer
p	grid cell
s	represents conditions at the droplet surface or adjacent computational cell
t	conditions associated with time
,	partial differentiation with respect to the variable followed by it

II. Introduction

The gas-turbine combustor flows are often characterized by a complex interaction between various physical processes associated with the interaction between the liquid and gas phases, droplet vaporization, turbulent mixing, heat release associated with chemical kinetics, radiative heat transfer associated with highly absorbing and radiating species, among others. The rate controlling processes often interact with each other at various disparate time and length scales. In particular, turbulence plays an important role in determining the rates of mass and heat transfer, chemical reactions, and liquid phase evaporation in many practical combustion devices. Most of the turbulence closure models for reactive flows have difficulty in treating nonlinear reaction rates.[1-2] The use of assumed shape PDF methods was found to provide reasonable predictions for pattern factors and NO_X emissions at the combustor exit. However, their extension to multi-scalar chemistry becomes quite intractable. The solution procedure based on the modeled joint composition PDF transport equation has an advantage in that it treats the nonlinear reaction rates without any approximation. This approach holds the promise of modeling various important combustion phenomena relevant to practical combustion devices such as flame extinction and blow-off limits, and unburnt hydrocarbons (UHC), CO, and NO_X predictions.[2]

However, in the PDF transport equation, all of the composition variables are present as independent variables in addition to the space and time variables. Because of the large dimensionality of a joint scalar PDF transport equation, a deterministic solution becomes impractical.[2] However, Monte Carlo methods are better suited over other numerical methods because of the advantage that the computational effort rises only linearly with the dimensionality of the PDF. But they tend to be computationally very time consuming and require a large computer memory for application to 3D flows. However, the success of any numerical methodology used in the study of practical combustion flows depends not only on the modeling and numerical accuracy considerations; but its applicability would be dictated mainly by the available computer memory and turnaround times afforded by the present-day computers.

Other than some simple cases, there is a limited experience with the application of the PDF method based on the solution of the PDF transport equation to the calculations involving three-dimensional practical combustion flows.[3] With the aim of demonstrating its viability to the modeling of practical combustion flows, we have undertaken the task of extending this technique to: (1) the modeling of sprays in order to facilitate simulation of gas-turbine combustion flows, (2) parallel computing in order to facilitate large-scale combustor computations, and (3) unstructured grids in order to facilitate computations on complex combustion geometries. Also, several numerical techniques are outlined for overcoming some of the high computer time and storage limitations associated with the Monte Carlo simulation of practical combustor flows.

Some of our previous work on the Monte Carlo simulation could be found in Refs. 4 to 7. Initially, the emphasis of our work was on extending the joint scalar Monte Carlo PDF method to the modeling of compressible reacting flows.[4] The Monte Carlo solver was used in conjunction with several density-based codes for the mean-velocity and turbulence fields. Several averaging procedures introduced in Refs. 4 and 5 proved to be useful in providing smooth Monte Carlo solutions to the CFD solver. The PDF method provided favorable results when applied to several supersonic diffusion flames.[4–5]

Later on this approach was further extended to the modeling of spray flames and parallel computing.[6] This method combined the novelty of the PDF method with the ability to run on parallel architectures. This algorithm was implemented on the Cray T3D at NASA Lewis Research Center, a massively parallel computer, with an aggregate of 64 Processor Elements (PEs). The computer code was written in Cray MPP (Massively Parallel Processing) Fortran. The application of this method to both open as well as confined axisymmetric swirl-stabilized spray flames showed good agreement with the measured data.[6] Preliminary estimates indicated that it was well within modern parallel computer's capacity to do a realistic gas-turbine combustor simulation on grid sizes of 125,000 nodes and a total of 12.5 million Monte Carlo particles with reasonable turnaround times.

It is well known that considerable effort usually goes into generating structured grid meshes for gridding up practical combustor geometries which tend to be very complex in shape and configuration. The grid generation time could be reduced considerably by making use of existing automated unstructured grid generators.[8] With the aim of advancing the current multi-dimensional computational tools used in the design of advanced technology combustors, we have recently extended this technique to unstructured grids following the guidelines established for the development of the National Combustion Code (NCC).[7] NCC is being developed in the form of a collaborative effort between NASA LeRC, aircraft engine manufacturers, and several other government agencies and contractors.[9] Some of the salient features of our work in Ref. 7 are summarized below:

(1) The scalar Monte Carlo PDF solver has been extended in a cell-centered, finite volume context to mixed unstructured grid elements of triangular, quadrilateral, tetrahedron, wedge, and hexahedron elements.

(2) The PDF code was rewritten in Fortran 77 with PVM calls for parallel computing which enables the computations to be performed with equal ease on both massively parallel computers as well as workstation clusters.

(3) The PDF module was coupled with Pratt and Whitney's CORSAIR[10] - an unstructured flow solver, and LSPRAY[11] - a Lagrangian spray solver, which were selected to be the integral components of the NCC cluster of modules. LSPRAY was developed for application with unstructured grids and parallel computing.

(4) Our experience has shown that by introducing the concept of local time-stepping into the Monte Carlo PDF computations, we were able to avoid the occurrence of the so-called frozen condition[3] and, thereby, able to compute the solution with the use of a relatively fewer number of stochastic particles in flows characterized by recirculation, swirl, and boundary layers.

(5) The effect of temperature variation was taken into account in the evaluation of both temperature and specific enthalpy of a gaseous mixture.

(6) The PDF solver receives the mean-velocity and turbulence fields from the flow solver and the source terms arising from the liquid-phase contribution from the spray solver. It in turn provides the species and temperature solution to the spray and CFD solvers.

The PDF method seems to provide favorable agreement when applied to several supersonic diffusion flames as well as several other swirl-stabilized spray flames.[4-7]

III. Composition Joint PDF Equation

The transport equation for the density-weighted joint PDF of the compositions, $\tilde{p}$, is:

$$
\begin{gathered}
[\bar{\rho}\tilde{p}]_{,t} + [\bar{\rho}\tilde{u}_i\tilde{p}]_{,x_i} + [\bar{\rho}w_\alpha(\underline{\psi})\tilde{p}]_{,\psi_\alpha} = \\
\{Mean\ convection\} \quad \{Chemical\ reactions\} \\
-[\bar{\rho} < u_i'' \mid \underline{\psi} > \tilde{p}]_{,x_i} - [\bar{\rho} < \frac{1}{\rho} J^{\alpha}_{i,x_i} \mid \underline{\psi} > \tilde{p}]_{,\psi_\alpha} \\
\{Turbulent\ convection\} \quad \{Molecular\ mixing\} \\
-\ [\bar{\rho} < \frac{1}{\rho} s_\alpha \mid \underline{\psi} > \tilde{p}]_{,\psi_\alpha} \qquad (1) \\
\{Liquid - phase\ contribution\}
\end{gathered}
$$

where

w_α	=	chemical source term for the α-th composition variable,
$< u_i'' \mid \underline{\psi} >$	=	conditional average of Favre velocity fluctuations,
$< \frac{1}{\rho} J^{\alpha}_{i,x_i} \mid \underline{\psi} >$	=	conditional average of scalar dissipation, and
$< \frac{1}{\rho} s_\alpha \mid \underline{\psi} >$	=	conditional average of spray source terms.

The terms on the left hand side of the above equation could be evaluated without any approximation but the terms on the right hand side of the equation require modeling. The first term on the right represents transport in physical space due to turbulent convection.[2] Since the joint PDF, $\tilde{p}$, contains no information on velocity, the conditional expectation of $< u_i'' \mid \underline{\psi} >$ needs to be modeled. It is modeled based on a gradient-diffusion model with information supplied on the turbulent flow field from the flow solver.[2]

$$- < u_i'' \mid \underline{\psi} > \tilde{p} = \Gamma_\phi \tilde{p}_{,x_i} \tag{2}$$

The second term on the right hand side represents transport in the scalar space due to molecular mixing. A mathematical description of the mixing process is rather complicated and the interested reader is refered to Ref. 2. Molecular mixing is accounted for by making use of the relaxation to the ensemble mean submodel[1] as it seemed to provide rather satisfactory results.[12]

$$< \frac{1}{\rho} J^{\alpha}_{i,x_i} \mid \underline{\psi} >= -C_\phi \omega (\phi_\alpha - \bar{\phi}_\alpha) \tag{3}$$

The third term on the right hand side represents the contribution from the the spray source terms. The conditional average is modeled based on the average values of species and enthalpy:

$$< \frac{1}{\rho} s_\alpha \mid \underline{\psi} >= \frac{1}{\bar{\rho} \Delta V} \sum n_k m_k (\epsilon_{\alpha s} - \phi_\alpha) \tag{4}$$

where $\phi_\alpha = Y_\alpha, \alpha = 1, 2, ..., s = \sigma - 1$

$$< \frac{1}{\rho} s_\alpha \mid \underline{\psi} >= \frac{1}{\bar{\rho} \Delta V} \sum n_k m_k (-l_{k,eff} + h_{ks} - \phi_\alpha) \tag{5}$$

where $\phi_\sigma = h$.

where $\epsilon_{\alpha s}$ is a mass fraction of the evaporating species at the droplet surface. Here we assumed that the spray source terms could be evaluated independent of the fluctuations in the gas phase compositions of species and enthalpy. Eqs. 4 to 5 represent the modeled representation for the conditional averages of the spray contribution to the PDF transport equation.

IV. Solution Algorithm

In order to facilitate the integration of the Monte Carlo PDF method in a finite-volume context, the volume integrals of convection and diffusion in Eq. (1) were first recast into surface integrals by means of a Gauss's theorem.[3] Partial integration of the PDF transport equation would yield:

$$\tilde{p}_p(\underline{\psi}, t+\Delta t) = (1 - \frac{c_p \Delta t}{\bar{\rho}\Delta V})\tilde{p}_p(\underline{\psi}, t) + \sum_n \frac{c_n \Delta t}{\bar{\rho}\Delta V}\tilde{p}_n(\underline{\psi}, t)$$

$$- \Delta t[w_\alpha(\underline{\psi})\tilde{p}]_{,\psi_\alpha} - \Delta t[< \frac{1}{\rho} J^\alpha_{i,x_i} \mid \underline{\psi} > \tilde{p}]_{,\psi_\alpha} - \Delta t[< \frac{1}{\rho} s_\alpha \mid \underline{\psi} > \tilde{p}]_{,\psi_\alpha} \qquad (6)$$

with subscript n refers to the nth-face of the computational cell. The coefficient c_n represents the transport by convection and diffusion through the nth face of the computational cell, p. The convection/diffusion coefficients in the above equation are determined by one of the following two expressions:

$$c_n = \Gamma_\phi(\frac{2\underline{a}_n \cdot \underline{a}_n}{\Delta V_p + \Delta V_s}) + max[0, -\bar{\rho}\underline{a}_n \cdot \underline{u}_n]$$

$$c_n = max[|0.5\bar{\rho}\underline{a}_n \cdot \underline{u}_n|, \Gamma_\phi(\frac{2\underline{a}_n \cdot \underline{a}_n}{\Delta V_p + \Delta V_s})\,] - 0.5\bar{\rho}\underline{a}_n \cdot \underline{u}_n$$

and

$$c_p = \sum_n c_n$$

In both the above expressions for c_n, a cell-centered finite-volume derivative is used to describe the viscous fluxes; but an upwind differencing scheme is used for the convective fluxes in the first expression and a hybrid differencing scheme in the second.

Numerical Method Based on Approximate Factorization

The transport equation is solved by making use of an approximate factorization scheme.[2] Eq. 6 can be recast as

$$\tilde{p}_p(\underline{\psi}, t+\Delta t) =$$

$$(I + \Delta t R)((I + \Delta t S)(I + \Delta t M)(I + \Delta t T)\tilde{p}_p(\underline{\psi}, t) + O(\Delta t^2) \qquad (7)$$

where I represents the unity operator and T, M, S, and R denote the operators associated with spatial transport, molecular mixing, spray, and chemical reactions, respectively. The operator is further split into a sequence of intermediate steps:

$$\tilde{p}^\star_p(\underline{\psi}, t) = (I + \Delta t T)\tilde{p}_p(\underline{\psi}, t) \qquad (8)$$

$$\tilde{p}_p^{\star\star}(\underline{\psi}, t) = (I + \Delta t M)\tilde{p}_p^{\star}(\underline{\psi}, t) \tag{9}$$

$$\tilde{p}_p^{\star\star\star}(\underline{\psi}, t) = (I + \Delta t S)\tilde{p}_p^{\star\star}(\underline{\psi}, t) \tag{10}$$

$$\tilde{p}_p(\underline{\psi}, t + \Delta t) = (I + \Delta t R)\tilde{p}_p^{\star\star\star}(\underline{\psi}, t) \tag{11}$$

The operator splitting method provides the solution for the transport of $\tilde{p}$ by making use of a Monte Carlo technique. In the Monte Carlo simulation the dentity weighted PDF at each grid cell is represented by an ensemble of N_m stochastic elements where the ensemble-averaged PDF over N_m delta functions replaces the average based on a continuous PDF.[2]

$$\tilde{p}_{pm}(\psi) =< \tilde{p}_p(\psi) >= \frac{1}{N_m}\sum_{n=1}^{N_m} \delta(\psi - \phi^n) \tag{12}$$

The discrete PDF $\tilde{p}_{pm}(\psi)$ is defined in terms of N_m sample values of ϕ^n, $n = 1, 2, 3 \ldots N_m$. The statistical error in this approximation is proportional to $N_m^{1/2}$.

Using the operator splitting method, the solution for the PDF transport equation is obtained sequentially according to the intermediate steps given by Eqs. 8 to 11.

Convection/Diffusion Step

The first step associated with convection/diffusion is given by:

$$\tilde{p}_p^{\star}(\underline{\psi}, t) = (I + \Delta t T)\tilde{p}_p(\underline{\psi}, t) =$$

$$(1 - \frac{c_p \Delta t}{\bar{\rho}\Delta V})\tilde{p}_p(\underline{\psi}, t) + \sum_n \frac{c_n \Delta t}{\bar{\rho}\Delta V}\tilde{p}_n(\underline{\psi}, t) \tag{13}$$

This step is simulated by replacing a number of particles (= the nearest integer of $\frac{c_n \Delta t N_m}{\bar{\rho}\Delta V}$) at $\phi_P(t)$ by randomly selected particles at $\phi_n(t)$.

Numerical Issues Associated With Fixed Versus Variable Time Step

It is obvious from the above equation that a necessary criterion for stability requires satisfaction of $\frac{c_p \Delta t}{\bar{\rho}\Delta V} < 1$. This restriction needs to be satisfied for all grid nodes at the same time. However, this criterion tends to be too restrictive for applications involving complex combustor chamber flows where resolution considerations require concentration of the grid lines more in certain regions of the flowfield than the others. For example, more grid lines are clustered in regions where boundary layers are formed. The time step as determined by a limited region in the computational domain can lead to a frozen condition in some other nodes where there may be no elements for shifting across the neighboring cells. In order to avoid this frozen condition, the following criterion

$$\frac{c_n \Delta t N_m}{\bar{\rho}\Delta V} > 1$$

has to be satisfied at all grid nodes; but such a restriction could invariably lead to a prohibitively large cpu time. Scheurlen et al.[3] were the first ones to recognize the limitations associated with the use of a fixed time step in the Monte Carlo PDF computations.

However, our experience has shown that this problem can be overcome by introducing the concept of local time-stepping which is a convergence improvement technique widely used in many of the steady-state CFD computations. In this approach, the solution is advanced at a variable time step for different grid nodes. In our present computations, it is determined based on

$$\Delta t = min(C_{tf}\Delta t_f, \frac{\rho \Delta V}{C_t(c_n + S_{mlc})})$$

where C_{tf} and C_t are constants and were assigned the values of 4 and 2.5, respectively, Δt_f is the local time step obtained from the flow (CORSAIR) module, and $S_{mlc} = \sum n_k m_k$. The time step is chosen such that it permits transfer of enough particles across the boundaries of the neighboring cells while ensuring that the time step used in the PDF computations does not deviate very much from the time step used in the flow solver.

Molecular Mixing Step

The second step associated with molecular mixing is given by

$$\frac{d\phi_\alpha}{dt} = -C_\phi \omega(\phi_\alpha - \bar{\phi}_\alpha) \quad (14)$$

The solution for this equation is updated by:

$$\phi_\alpha^{\star\star} = \phi_\alpha^\star + (\phi_\alpha^\star - \bar{\phi}_\alpha^\star)e^{-C_\phi \omega \Delta t} \quad (15)$$

where $\omega = \epsilon/k$, and C_ϕ was assigned a value of 1.

Spray Step

The third step associated with the spray contribution is given by

$$\frac{d\phi_\alpha}{dt} = \frac{1}{\bar{\rho}\Delta V}\sum n_k m_k(\epsilon_\alpha - \phi_\alpha) \tag{16}$$

where $\phi_\alpha = Y_\alpha, \alpha = 1, 2, ..., s = \sigma - 1$

$$\frac{d\phi_\alpha}{dt} = \frac{1}{\bar{\rho}\Delta V}\sum n_k m_k(-l_{k,eff} + h_{ks} - \phi_\alpha) \tag{17}$$

where $\phi_\sigma = h$.

The solution for the above equations is upgraded by a simple explicit scheme:

$$\phi_\alpha^{***} = \epsilon_\alpha \frac{\Delta t \sum n_k m_k}{\bar{\rho}\Delta V} + \phi_\alpha^{**}(1 - \frac{\Delta t \sum n_k m_k}{\bar{\rho}\Delta V}) \tag{18}$$

where $\alpha \leq \sigma - 1$

$$\phi_\alpha^{***} = \frac{\Delta t \sum n_k m_k}{\bar{\rho}\Delta V}(-l_{k,eff} + h_{ks}) + \phi_\alpha^{**}(1 - \frac{\Delta t \sum n_k m_k}{\bar{\rho}\Delta V}) \tag{19}$$

where $\alpha = \sigma$.

After a new value for enthalpy is updated, temperature is determined iteratively from the following equation:

$$h = \sum_{i=1}^{N_{\sigma-1}} y_i h_i \tag{20}$$

where

$$h_i = h_{fi}^o + \int_{T_{ref}}^{T} C_{pi} y_i dT,$$

$$C_{pi} = \frac{R_u}{W_i}(A_{1i} + A_{2i}T + A_{3i}T^2 + A_{4i}T^3 + A_{5i}T^4),$$

h_{fi}^o is the heat of formation of ith species, and R_u is the universal gas constant.

Reaction Step

Finally, the fourth step associated with chemical reactions is given by:

$$\frac{d\phi_\alpha}{dt} = -\nu_f \frac{W_f}{\rho} A(\frac{\rho\phi_f}{W_f})^{0.25}(\frac{\rho\phi_o}{W_o})^{1.25} e^{-(\frac{E_a}{T})} \tag{21}$$

where $\phi_\alpha = Y_f$.

$$\frac{d\phi_\alpha}{dt} = -\nu_o \frac{W_o}{\rho} A (\frac{\rho \phi_f}{W_f})^{0.25} (\frac{\rho \phi_o}{W_o})^{1.25} e^{-(\frac{E_a}{T})} \tag{22}$$

where $\phi_\alpha = Y_o$.

$$\frac{d\rho\phi_\alpha}{dt} = 0 \tag{23}$$

where $\phi_\alpha = h$.

The numerical solution for Eqs 20-23 is integrated by an implicit Euler scheme.[13] The resulting non-linear algebraic equations are solved by the method of quasi-linearization.[14]

Details of combustion chemistry

Combustion is modeled by finite-rate kinetics with a single step global mechanism of Westbrook and Dryer[15] for n-heptane. This global combustion model is reported to provide adequate representation of temperature histories in flows not dominated by long ignition delay times. For example, the overall reaction representing the oxidation of the n-heptane fuel is given by

$$\begin{aligned} C_7H_{16} + 11(O_2 + 3.76N_2) \rightarrow \\ 7CO_2 + 8H_2O + 41.36N_2 \end{aligned} \tag{24}$$

Because of the constant Schmidt number assumption made in the PDF formulation, based on atomic balance of the constituent species, the mass fractions of N_2, CO_2, and H_2O can be shown to be related to the mass fractions of O_2 and C_7H_{16} by the following expressions:

$$\begin{aligned} y_{H_2O} &= K_2 - K_1K_2y_{O_2} - K_2y_{C_7H_{16}} \\ y_{CO_2} &= K_2K_3 - K_1K_2K_3y_{O_2} - K_2K_3y_{C_7H_{16}} \\ y_{N_2} &= 1 - K_2 - K_2K_3 - y_{O_2}(1 - K_1K_2 - K_1K_2K_3) - \\ & \quad y_{C_7H_{16}}(1 - K_2 - K_2K_3) \end{aligned} \tag{25}$$

where $K_1 = 4.29$, $K_2 = 0.08943$, and $K_3 = 2.138$.

Using Eq. 25 results in considerable savings in computational time as it reduces the number of variables in the PDF equation from five (four species and one energy) to three (two species and one energy).

Revolving Time-Weighted Averaging

It is noteworthy that although local time stepping seems to overcome some of the problems associated with the PDF computations, the application of the Monte Carlo method requires the use of a large number of particles because the statistical error associated with the Monte Carlo Method is proportional to the square root of N_m which makes the use of the Monte Carlo method computationally very time consuming. However, a revolving averaging procedure used in our previous work[4] seems to alleviate the need for using a large number of stochastic particles, N_m, in any one given time step. In this averaging scheme, the solution provided to the CFD solver is based on an average of all the particles present over the last N_{av} time steps instead of an average based solely on the number of particles present in any one single time step. This approach seemed to provide smooth Monte Carlo solutions to the CFD solver and, thereby, improving the convergence of the coupled CFD and Monte Carlo computations. The reason for improvement could be attributed to the average being based on $N_{av}N_m$ particles instead of N_m. Here, it is assumed that the solution contained in different time steps to be statistically independent of each other.

V. Parallelization

There are several issues associated with the parallelization of the PDF computations. The goal of the parallel implementation is to extract maximum parallelism so as to minimize the execution time for a given application on a specified number of processors.[16] Several types of overhead costs are associated with parallel implementation which include data dependency, communication, load imbalance, arithmetic, and memory overheads. Arithmetic overhead is referred to the extra arithmetic operations required by the parallel implementation and memory overhead refers to the extra memory needed. Excessive memory overhead reduces the size of a problem that can run on a given system and the other overheads result in performance degradation.[16] Any given application usually consists of several different phases that must be performed in certain sequential order. The degree of parallelism and data dependencies associated with each of the subtasks can vary widely.[16] The goal is to achieve maximum efficiency with a reasonable programming effort.

In our earlier work, we discussed the parallel implementation of a spray algorithm developed for the structured grid calculations on a Cray T3D.[6] These computations were performed in conjunction with the application of the Monte Carlo PDF method to spray flames. The parallel algorithm made use of the shared memory constructs exclusive to Cray MPP (Massively Parallel Processing) Fortran and the computations showed a reasonable degree of parallel performance when they were performed on a NASA LeRC Cray T3D with the number of processors ranging between 8 to 32. Later on, the extension of this method to unstructured grids and parallel computing in

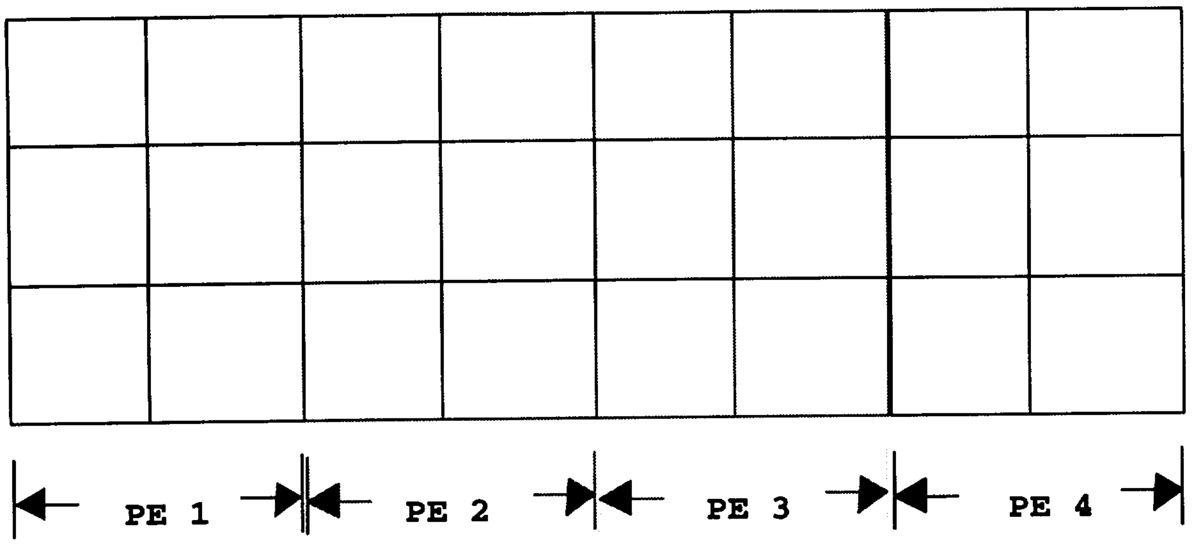

Fig. 1 An illustration of the parallelization strategy employed in EUPDF.

Fortran 77 with PVM calls was reported in Ref. 7. The Fortran 77 version offers greater computer platform independence. In this section, we only highlight some important aspects of parallelization.

In the domain decomposition employed, the domain of computation is simply divided into n-parts of equal size and each part is solved by a different processor. Fig. 1 illustrates a simple example of the domain decomposition strategy adopted for the gas-phase computations where the total domain is simply divided equally amongst the available computer processing elements (PEs). In this case, we assumed the number of available PEs to be equal to four.

At the beginning of the computation, all the information that is needed from the adjoining cells in computing the cell-face variables at the boundaries of the interface separating two neighboring processor-domains is obtained from the appropriate processors and stored in ghost cells. And interprocessor communications are performed only at the beginning of integration step as the need arises. With the domain decomposition adopted, all the stages of a single PDF integration step involving the spatial transport, molecular mixing, spray, and chemical kinetics lend themselves perfectly to parallel computing. Therefore, the Monte Carlo simulation is ideally suited for parallel computing and the run time could be considerably minimized by performing

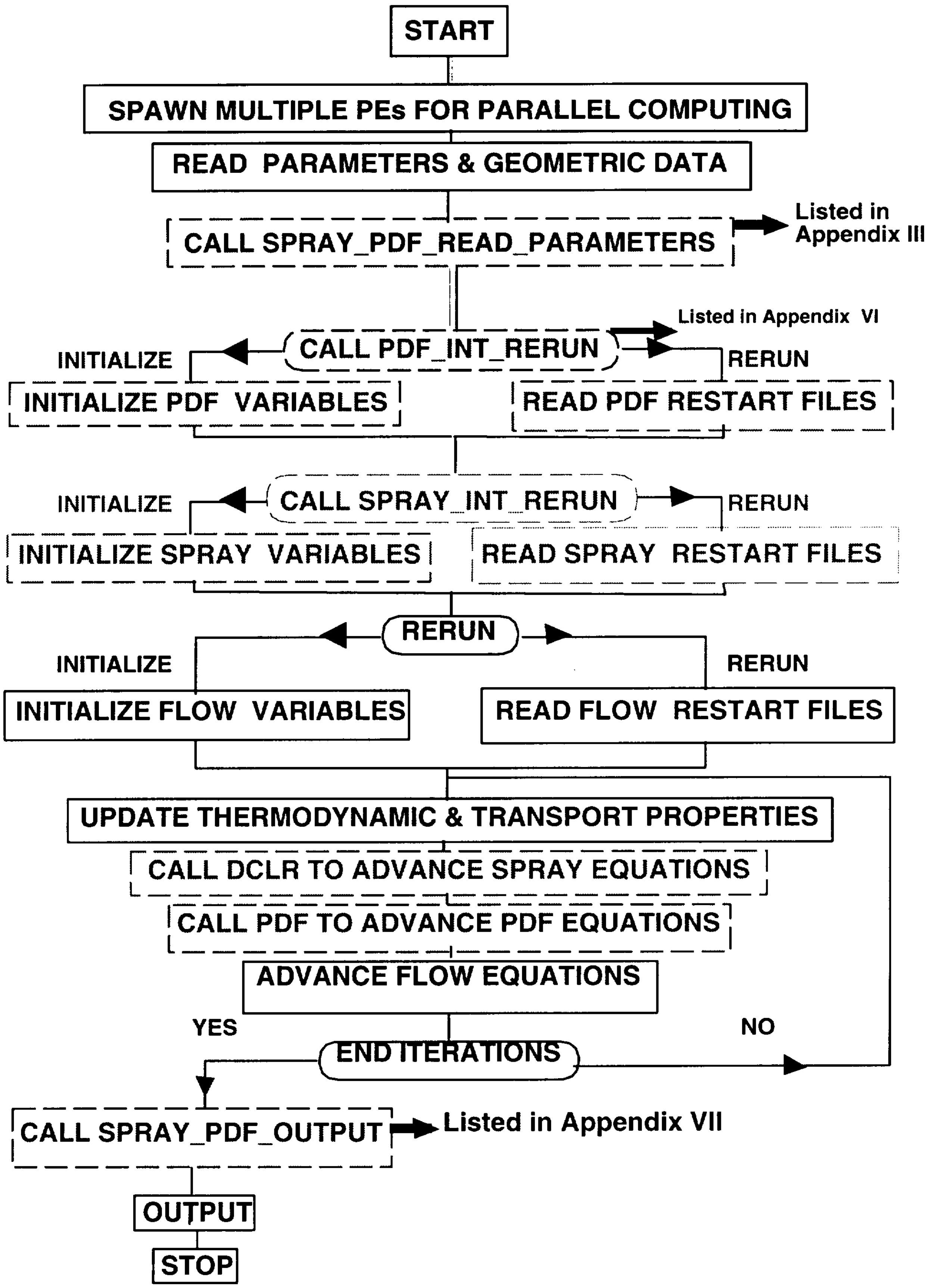

Fig. 2 The overall flow structure of the combined flow, LSPRAY, ad EUPDF solvers.

the computations on a massively parallel computer.

VI. Details of the Coupling With the Flow & Spray Solvers

The PDF module is designed so that it could easily be coupled with any of the existing unstructured-grid flow and spray solvers. If geometric grid parameters - e.g. area vectors, grid connectivity, etc., were supplied separately, it could even be coupled with any of the existing structured-grid flow solvers. However, the present release of the code relies on the other modules of NCC for obtaining that information.

The structure of the PDF solver is so designed that only a minimal amount of code modifications needs to be made within the flow and spray solvers for their coupling with EUPDF. The present version of the module relies entirely on the use of the Fortran common blocks for information exchange between the various modules. Even this reliance should entail only few changes to be made within the PDF code for linkage with different solvers.

The coupling issues could be understood better through the use of a flow chart shown in Fig. 2. The chart contains several blocks - some shown in solid lines and others in dashed lines. The ones in solid blocks represent the flow chart that is typical of most flow solvers. The ones in dashed blocks represent the coupling required for adding the spray and PDF solvers. The details on the spray blocks are not provided in this report as they could be found in the separate reference.[11] It should be borne in mind that the PDF solver could be run without the spray solver and vice-versa as they are independent.

The flow chart for a typical flow solver begins by calling several routines - some for initiating the established PVM protocol for parallel computing and the others for spawning children of the same processes so that the computations could be performed simultaneously on various PEs participating in the parallel computing environment. It is followed by a routine to read various initial parameters. The geometric data could be either read directly or created by the inclusion of appropriate calling routines needed for grid generation. Then, the initial conditions for the flow variables need to be either specified or read from the restart files if it is a rerun. The thermodynamic and transport properties are then updated before advancing the flow equations over a series of time steps until the desired number of iterations are reached. Finally, the program is terminated after writing the output data on the separate restart and standard files.

The coupling starts with the addition of a calling routine - **spray_pdf_read_parameters** - to read the spray and PDF control parameters followed by calls to the restart or initialization routines: **pdf_int_rerun** followed by **spray_int_rerun**. Then, calls to **dclr and pdf** were made in order to advance the spray and PDF equations in a sequential order before advancing the flow equations. It should be borne in mind that if the PDF solver is invoked, the thermodynamic and transport properties would be evaluated by the routines contained within the PDF solver instead of the ones contained in the flow solver. Finally, **spray_pdf_output** and **outpdf2** routines are included for outputting the PDF and spray data on appropriate restart and standard files.

Appendix I contains a listing of all of the Fortran subroutines and functions used in EUPDF.

A brief description of each one of these routines is provided in Appendix II.

Appendix III contains the listing of a subroutine which is used for reading some of the control and other associated parameters involving LSPRAY and EUPDF solvers.

Appendix IV contains a list of some Fortran variables used by EUPDF.

Appendix V contains a list of the geometric variables used by EUPDF which are currently supplied by the flow code of NCC.

Appendix VI contains a subroutine for initialization and restart of the PDF computations. For the case of initialization, the PDF variables are initialized based on the solution provided by the flow solver. For the case of restart, the the PDF variables are read from a previous solution.

Appendix VII contains the listing of subroutines **spray_pdf_output** and **outpdf2** which are for writing output data from LSPRAY and EUPDF codes on separate standard and restart files.

Appendix VIII contains an example of the partial listings of code initiation for coupling LSPRAY and EUPDF with a gas flow solver.

Appendix IX contains a listing of all subroutines and functions used in the evaluation of the thermodynamic and transport Properties which are used not only in EUPDF but also in the flow solver as well as in LSPRAY. These subroutines include **rctinp**, **props_ther**, **props_tran**, **get_kt_and_cp_loc_mcs**, **find_inverse_molecular_weight_mcs**, and **find_h_mcs**.

The last appendix provides an example of the summary of the CPU times taken by CORSAIR and EUPDF for the case of a confined swirl-stabilized spray flame when the computations were performed on a LACE cluster at NASA LeRC.

VII. Concluding Remarks

A Monte Carlo technique has been outlined for the computation of spray flames on unstructured grids with parallel computing. The method outlines several techniques designed to overcome some of the high computer time and storage limitations associated with the Monte Carlo simulation of practical combustor flows. The viability of the present method for its application to the modeling of practical combustion devices is demonstrated through the ease with which grids could be generated for complex combustor geometries by the use of automated unstructured mesh generators and the ability to run the computations on massively parallel computers.

VIII. Acknowledgements

His sincere appreciation goes to several members of the NASA LeRC management for their past and continued support in letting him extend his work on the scalar Monte Carlo PDF method to different applications involving sprays, parallel computing, and unstructured grids.

IX. References

1. Borghi, R., "Turbulent Combustion Modeling," Prog. Energy Combust. Sci., Vol. 14, pp. 245-292, 1988.
2. Pope, S.B., "PDF Methods for Turbulent Reactive Flows," Prg. Energy Combust. Sci., Vol. 11, pp. 119-192, 1985.
3. Scheurlen, M., Noll, B., and Wittig, S., "Application of Monte Carlo Simulation For Three-Dimensional Flows," In AGARD-CP-510, CFD Techniques For Propulsion Applications, February 1992.
4. A.T. Hsu, Y.-L.P. Tsai, and M.S. Raju, "A Probability Density Function Approach for Compressible Turbulent Reacting Flows," AIAA Journal, Vol. 32, No. 7, pp. 1407-1415, 1994.
5. A.T. Hsu, M.S. Raju, and A.T. Norris, "Application of a pdf Method to Compressible Turbulent Reacting Flows," AIAA 94-0781, AIAA 32nd Aerospace Sciences Meeting, Reno, Nevada, January 1994.
6. Raju, M.S., "Application of Scalar Monte Carlo Probability Density Function Method For Turbulent Spray Flames," Numerical Heat Transfer, Part A, Vol. 30, pp. 753-777, 1996.
7. Raju, M.S., "Combined Scalar-Monte-Carlo-PDF/CFD Computations of Spray Flames on Unstructured Grids With Parallel Computing," AIAA Paper 97-2969, 33rd AIAA/ ASME/ SAE/ ASEE Joint Propulsion Conference, July 6-10, 1997/ Seattle, WA.
8. Shang, H.M., Chen, Y.S., Liaw, P., Shih, M.H., and Wang, T.S., "Numerical Modeling of Spray Combustion With an Unstructured-Grid Method," AIAA 95-2781, 31st AIAA/ ASME/ SAE/ ASEE Joint Propulsion Conference, July 10-12, 1995/San diego, CA.

9. Liu, N.S., and Stubbs, R.M., "Preview of National Combustion Code," AIAA 97-3114, 33rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference, July 6-9, 1997/Seattle, WA.

10. Ryder, R., "CORSAIR User's Manual: Version 1.0," SID: Y965, Pratt and Whitney Engineering, United Technologies Corporation, 25 January 1993.

11. Raju, M.S., "LSPRAY - a Lagrangian Spray Solver - User's Manual," NASA CR-97-206240, November 1997.

12. Correa, S.M. "Development and Assessment of Turbulence-Chemistry Models in Highly Strained Non-Premixed Flames," AFOSR/NA Contractor Report, 110 Duncan Avenue, Bolling AFB, DC 20332-0001, 31 October, 1994.

13. Anderson, D.A., Tannehill, J.C., and Fletcher, R.H., "Computational Fluid Mechanics and Heat Transfer," Series in Computational Methods in Mechanics and Thermal Sciences, Hemisphere Publishing Corporation, Washington D.C., U.S.A., 1984.

14. Raju, M.S., Liu, W.Q., and Law, C.K., " A Formulation of Combined Forced and Free Convection Past Horizontal and Vertical Surfaces," Int. J. Heat and Mass Transfer, Vol 27, pp. 2215-2224, 1984.

15. Westbrook, C.K. and Dryer, F.L., "Chemical Kinetic Modelling of Hydrocarbon Combustion," Progress in Energy and Combustion Science, Vol. 10, No. 1, 1984, pp. 1-57.

16. Ryan, J.S., and Weeratunga, S.K., "Parallel Computation of 3-D Navier-Stokes Flowfields for Supersonic Vehicles," AIAA 93-0064, 31st Aerospace Sciences Meeting and Exhibit, Reno, NV, 1993.

Appendix I

Table 1. A List of EUPDF Fortran Subroutines and Functions.

Number	Name of the routine	Code released with NCC/Description page	Code page/ User's manual
1.	chem2	25	
2.	coeffiv()	25	
3.	convec	25	
4.	find_h_mcs()	25	61
5.	find_inverse_molecular_weight_mcs()	26	60
6.	get_kt_and_cp_loc_mcs()	26	58
7.	molmix	26	
8.	outpdf1()	26	
9.	outpdf2()	26	46
10.	pdf	26	
11.	pdf_int_rerun	27	39
12.	prnt()	27	
13.	props_ther	27	55
14.	props_tran	27	57
15.	rctinp	27	53
16.	replac	27	
17.	spray	27	
18.	spray_pdf_output	27	43
19.	spray_pdf_read_parameters	27	29

Appendix II

Description of EUPDF Fortran Subroutines and Functions

1. subroutine chem2:

PURPOSE: This subroutine integrates the chemical kinetics step of the PDF solution which is especially tailored to solve a single step global mechanism of the Westbrook and Dryer for n-heptane oxidation.

2. subroutine coeffiv(axyzp,d):

PURPOSE: Computes convection/diffusion coefficients based on a cell-centered finite-volume derivative for diffusion and a upwind or hybrid differencing scheme for the convection.

3. subroutine convec:

PURPOSE: This is the controlling routine for the integration of the convection/diffusion step.

(1) Computes the convection/diffusion coefficients by calling the subroutine coeffiv.

(2) Converts the decimal numbers into integer numbers which represent the number of particles to be transferred across the neighboring cells.

(3) Loads the random numbers into appropriate arrays which are used in determining the particles that need to be transferred across neighboring cells.

(4) Moves particles across different neighboring cells based on the computed convection/diffusion coefficients by calling the subroutine replac.

4. function find_h_mcs(element,centroid):

PURPOSE: Computes specific enthalpy at a nodal point of the computational grid.

5. **function find_inverse_molecular_ weight_mcs(element,centroid):**

PURPOSE: Computes inverse of the molecular weight of a gaseous mixture at a nodal point of the computational grid.

6. **subroutine get_kt_and_cp_loc_mcs(ijk,centroid):**

PURPOSE: Computes specific heat and thermal conductivity of a gaseous mixture at a nodal point of the computational grid.

7. **subroutine molmix:**

PURPOSE: This is a routine for computing transport in scalar space due to molecular mixing. It uses the relaxation to the ensemble-mean molecular mixing model.

8. **subroutine outpdf1(ncyc):**

PURPOSE: Calculates the residuals for the PDF solution.

9. **subroutine outpdf2(ncyc):**

PURPOSE: Outputs the PDF solution to restart files.

10. **subroutine pdf:**

PURPOSE: This is the main controlling routine for the Monte Carlo PDF solver. It updates the pdf solution through a series of calls to different subroutines and returns control back to the calling routine. It also performs the following functions:

(1) controls interprocessor communications.

(2) calculates the average value of scalars over particles.

(3) transfers the time-averaged PDF solution into corresponding arrays of the conventional CFD solver.

(4) computes thermodynamic and transport properties.

(5) computes the residual terms.

11. **subroutine pdf_int_rerun:**

```
PURPOSE: This routine is used either to initialize the Monte
         Carlo PDF computations or to restart the PDF computations
         from the output of a previous PDF calculation.
```

12. **subroutine prnt(aass,phig,nbc,nnode,np,ipid, nstart,nend,iul):**

```
PURPOSE:    I/O print out.
```

13. **subroutine props_ther:**

```
PURPOSE: This routine computes enthalpy, specific heat, and
         density.
```

14. **subroutine props_tran:**

```
PURPOSE: This routine computes transport properties.
```

15. **subroutine rctinp:**

```
PURPOSE: Initializes parameters used in the thermodynamic
         and transport property evaluation as well as in
         the chemical kinetics scheme.
```

16. **subroutine replac:**

```
PURPOSE: Moves particles across different neighboring cells
         during the convection/diffusion step.
```

17. **subroutine spray:**

```
PURPOSE: Updates PDF solution associated with the contribution
         arising from the liquid-phase source terms.
```

18. **subroutine spray_pdf_output:**

```
PURPOSE: This routine writes output data from EUPDF & LSPRAY
         computations to restart and standard-output
         files.
```

19. **subroutine spray_pdf_read_parameters:**

```
PURPOSE: This routine reads controlling parameters associated
         with the EUPDF and LSPRAY solvers. Based on the controlling
         parameters read, it might invoke an initialization routine
         of the EUPDF solver which is needed in the thermodynamic &
         transport properties evaluation.
```

A Subroutine Listing For the Read Parameters of LSPRAY and EUPDF

```
c
      subroutine spray_pdf_read_parameters
c
      include 'dcfslog.i'
      include 'dcfslog_rw.i'
c
c --------------------------------------------------
c ---
c
c PURPOSE: This routine reads controlling parameters associated
c          with the EUPDF and LSPRAY solvers. Based on the controlling
c          parameters read, it might invoke an initialization routine
c          of the EUPDF solver which is needed in the thermodynamic &
c          transport properties evaluation.
c
c FORM OF CALL: call spray_pdf_read_parameters
c
c
c ADDITIONAL I/O:
c
c    INPUT: spray_pdf_parameter_input
c
c    OUTPUT: None
c
c --------------------------------------------------
c ---
c lspray  controls turning on or off spray computations.
c lspray = .TRUE.  - turns on spray computations.
c        = .FALSE. - otherwise.
c
c ldread controls reading or not from restart files for
c        spray computations.
c ldread = .TRUE.  - restarts from previous runs.
c        = .FALSE. - starts from initial conditions.
c
c ispray_mod= This variable controls calls to the spray
c              solver. The spray solver is called once at
c              every ispray_mod times of CFD iterations.
```

```
c
c ipread = Assigned unit number for the file: liquid_input.
c idread = Assigned unit number for the file: liquid_results.
c idread2= Assigned unit number for the file: liquid_results_ini.
c idwrit = Assigned unit number for the file: liquid_results_new.
c idwrit2= Assigned unit number for the file: liquid_results_ini.
c
c ipdf controls turning on or off Monte Carlo PDF computations.
c ipdf = 0 turns off Monte Carlo PDF computations.
c      = 1 otherwise.
c
c ns serves two functions depending on whether ns has a
c    zero or non-zero value.
c ns = 0 starts the Monte Carlo PDF computations from
c        initial conditions.
c ns = a non-zero number restarts the computations from
c       a previous run. a non-zero number represents the
c       last iteration number of a previous run which is
c       used in the time-averaging scheme utilized in
c       the PDF computations.
c
c ipdf_mod  = This variable controls calls to the PDF
c              solver. The PDF solver is called once at
c              every ipdf_mod times of CFD iterations.
c
c ipdf_num = In a given cycle, the pdf solver is advanced over
c            a number of time steps given by ipdf_num.
c
c irea1 = Assigned unit number for the file: pdf_results.
c irea2 = Assigned unit number for the file: pdf_results_ave.
c iwri1 = Assigned unit number for the file: pdf_results.
c iwri2 = Assigned unit number for the file: pdf_results_ave.
c
c --------------------------------------------------
c
      open(unit=85,file='spray_pdf_parameter_input')
      read(85,*)
      read(85,*)lspray,ldread,ispray_mod
      read(85,*)
      read(85,*)ipread,idread,idwrit,idread2,idwrit2
      read(85,*)
      read(85,*)ipdf,ns,ipdf_mod,ipdf_num
      read(85,*)
      read(85,*)irea1,irea2,iwri1,iwri2
      close(unit=85)
c
c --------------------------------------------------
c ---
c
c Routine rctinp of the PDF solver provides initialization
c parameters used in the themodynamic and transport property
c evaluation as well as in the chemical kinetics scheme.
c
      if(ipdf.eq.1) then
       call rctinp
      endif
c
c --------------------------------------------------
c
      RETURN
      END
c
```

A Listing of Some Fortran Variables Used in EUPDF

```
c ------------------------------------------------------------
c ---
c
c arh = A    = constants in the equation used for representing the
c                chemical reaction rate.
c
c av1()    It contains an average based on nparti*nav particles.
c          It is an average employed in the revolving time-averaging
c          technique.
c
c avmany() It contains nav number of individual averages which
c          are computed over nparti.
c
c centroid = logical variable. If yes, it repesents the values at
c              the cell center. If not, it represents values at the
c              cell faces.
c
c cp0,..cp4   = coefficients of the polynomial used in determining the
c                 variable specific heat.
c
c dhform      = coefficients of the polynomial used in determining the
c                 specific enthalpy.
c
c dwm         = coefficients of the polynomial used in determining the
c                 inverse of the molecular weight of a mixture.
c
c eno2, epox,
c epfu,emno2,
c empox,empfu = constants used in the chemical
c                 kinetics solver.
c
c erh = E_a = constants in the equation used for representing the
c                chemical reaction rate.
c
c f1()    = fuel mass fraction at the center of the grid cell.
c f1bar() = fuel mass fraction at the cell face.
c
c f2()    = oxygen mass fraction at the center of the grid cell.
```

```
c f2bar() = oxygen mass fraction at the cell face.
c
c
c fluid() = logical variable. If yes, it repesents a fluid
c           element. If not, it represents a solid element.
c
c h()     = enthalpy at the center of the grid cell.
c hbar()  = enthalpy at the cell face.
c
c iconpc = store differences in array index convention used by the
c          flow and pdf codes.
c
c nav = N_{av} = number of time steps employed in the time-averaging technique.
c
c
c nparti  = N_m = number of particles per cell
c
c
c ns        = current iteration of the PDF solver.
c
c nscala  = sigma = number of scalars
c
c part1 contains  pdf solution from the previous time step
c
c part2 contains  pdf solution for the current time step
c
c pz = pressure
c
c ru = universal gas constant
c
c schmdi= Sc = Schmdit number (=1/0.7).
c
c t()     = temperature at the center of the grid cell.
c tbar()  = temperature at the cell face.
c
c w1,w2,w3    = constants resulting from some algebraic manipulation
c               which reduces the need for solving speciess equations
c               from 5 to 3.
c
c wmole       = W = molecular weight
c
c xnup, xnupp = nu = stoichiometric coefficients of reactants and
```

```
c                             products.
c
c
c ---------------------------------------------------------------
c
```

A Listing of Geometric Variables Used in LSPRAY and EUPDF

```
c ------------------------------------------------------------
c ---
c
c The EUPDF and LSPRAY modules expects the following inputs on
c the grid related information:
c
c nodes     = total number of the computational elements.
c nedge     = total number of cell faces in the computational domain.
c nfaces(i) = total number of faces of the element, i.
c
c edge(i,1) and edge(i,2) represent the adjacent elements
c         of the face, i, if the face, i, if the face happens
c         to be an interface between two elements. Otherwise
c         edge(i,1) represents the correponding boundary
c         condition identifier.
c
c face_to_edge(i,j) represents the face ID of the element, i,
c         and the face, j.
c
c c1(i,j) provides connectivity map. c1(i,j) = adjacent element
c         ID of the element, i, and the face, j, otherwise c1(i,j)
c         = boundary condition identifier on any boudary.
c
c vol(i) = volume of the element, i.
c
c areax(i), areay(i), and areaz(i) are the cartesian components
c         of the outward pointing area vector of the face, i.
c
c x1(i), y1(i), z1(i) are the cartesian components of the node
c        one of the element, i. Similarly, x2(i), y2(i), z2(i)
c        are for node 2 and so on.
c
c triangle(i) is .true. if i is a triangular element. Similarly,
c        quadrilateral(i), tetrahedron(i), and wedge(i) are logical
c        varibles representing other type of elements.
c
c axisymmetric is set to .true. for axisymmetric computations
c        otherwise it is .false. The axisymmetric computations are
```

```
c        performed by generating 3D elements from a 2D mesh with
c        an arc centered around the z coordinate, z=0.0. The angle
c        of the arc is defined by the variables, ARC, in radians
c        and THETA0, in degrees.
c
c ------------------------------------------------------------
c
```

A Subroutine Listing For EUPDF Code Initialization and Restart

```
c
      subroutine pdf_int_rerun
c
      include 'cfsparms.i'
      include 'cfsdt.i'
      include 'cfspert.i'
      include 'cfsconv.i'
      include 'cfstime.i'
      include 'cfsmimd.i'

      include 'cfsarea.i'
      include 'cfsnodes.i'
      include 'dcfslog.i'
      include 'dcfslog_rw.i'
      include 'cfsvars.i'
      include 'cfsprop.i'
      include 'cfsh.i'
c
c Include common blocks associated with PDF computations.
c
      include 'p3dpar.i'
      include 'p3dcom.i'
      include 'p3dave.i'
      include 'p3dpro.i'
c
c --------------------------------------------------
c ---
c
c PURPOSE: This routine is used either to initialize the Monte
c          Carlo PDF computations or to restart the PDF computations
c          from the output of a previous PDF calculation.
c
c FORM OF CALL: call pdf_int_rerun
c
c --------------------------------------------------
c
c --------------------------------------------------
c ---
c Initialize the computations if it is not a rerun.
```

```
c
c Initialize the PDF particle attributes as well as the
c averages to those provided by the flow solver.
c
      if(ipdf.eq.1.and.ns.eq.0) then
      do 39 ijk = 1,nodes
      avy1=f2(ijk)
      avy2=f1(ijk)
      avye=h(ijk)
      avyt=t(ijk)
      avmany(ijk,pox,2) = avy1
      avmany(ijk,pfu,2) = avy2
      avmany(ijk,pen,2) = avye
      avmany(ijk,pte,2) = avyt
      avmany(ijk,pox,1) = avy1
      avmany(ijk,pfu,1) = avy2
      avmany(ijk,pen,1) = avye
      avmany(ijk,pte,1) = avyt
 39   continue
      do 391 ijk = 1,nodes
      do 391 ip = 1,nparti
      part1(ijk,ip,pox) = avmany(ijk,pox,2)
      part1(ijk,ip,pfu) = avmany(ijk,pfu,2)
      part1(ijk,ip,pen) = avmany(ijk,pen,2)
      part1(ijk,ip,pte) = avmany(ijk,pte,2)
      part2(ijk,ip,pox) = avmany(ijk,pox,2)
      part2(ijk,ip,pfu) = avmany(ijk,pfu,2)
      part2(ijk,ip,pen) = avmany(ijk,pen,2)
      part2(ijk,ip,pte) = avmany(ijk,pte,2)
 391  continue
      do 321 ijk=1,nodes
      do 321 n=1,nav
      av1(ijk,pox,n) = avmany(ijk,pox,2)
      av1(ijk,pfu,n) = avmany(ijk,pfu,2)
      av1(ijk,pen,n) = avmany(ijk,pen,2)
      av1(ijk,pte,n) = avmany(ijk,pte,2)
 321  continue
      ns=nav
c
c Compute thermodynamic properties.
c
      call props_ther
```

```
c
c Compute transport properties.
c
      call props_tran
      endif
c
c --------------------------------------------------
c ---
c
c Read restart files. Also, initialize some other parameters
c of interest.
c
      if(ipdf.eq.1.and.ns.gt.nav) then
      itrecl=nparti*nscala*4+4
      open(unit=irea1,file='pdf_results',
     >access='direct',recl=itrecl,form='unformatted')
      itrecl=nscala*nav*4+nscala*2*4+4
      open(unit=irea2,file='pdf_results_ave',
     >access='direct',recl=itrecl,form='unformatted')
      do ijk=1,nodes
      irecord=ijk+nodes*(ipid-1)
      read(irea1,rec=irecord)
     >((part2(ijk,i1,i2),i1=1,nparti),i2=1,nscala)
      enddo
      do ijk=1,nodes
      irecord=ijk+nodes*(ipid-1)
      read(irea2,rec=irecord)
     >((av1(ijk,i1,i2),i1=1,nscala),i2=1,nav),
     >((avmany(ijk,i3,i4),i3=1,nscala),i4=1,2)
      enddo
c
      do 4000 ijk=1,nodes
      do 4000 i=1,nparti
      do 4000 k3=1,nscala
 4000 part1(ijk,i,k3)=part2(ijk,i,k3)
c
c Compute thermodynamic properties.
c
      call props_ther
c
c Compute transport properties.
c
```

```
      call props_tran
c
      endif
c
c ---
c --------------------------------------------------
c
      return
      end
c
```

Appendix VII

A Subroutine Listing For LSPRAY and EUPDF Data Output

```
c
      subroutine spray_pdf_output
c
      include 'cfsparms.i'
      include 'cfsdt.i'
      include 'cfspert.i'
      include 'cfsconv.i'
      include 'cfstime.i'
      include 'cfsmimd.i'

      include 'cfsarea.i'
      include 'cfsnodes.i'
      include 'cfsvars.i'
      include 'cfsprop.i'
      include 'cfsh.i'
c
c Include common blocks associated with spray and PDF computations.
c
      include 'dcfslog.i'
      include 'dcfslog_rw.i'
c
c Include common blocks associated with PDF computations.
c
      include 'p3dpar.i'
      include 'p3dcom.i'
      include 'p3dave.i'
      include 'p3dpro.i'
c
c Include common blocks associated with spray computations.
c
      include 'd3dpar.i'
      include 'd3dcom.i'
      include 'd3dinj.i'
      include 'd3dprl.i'
c
c ------------------------------------------------
c ---
c
```

```
c PURPOSE: This routine writes output data from PDF & spray
c          computations on a separate restart and standard
c          output files.
c
c FORM OF CALL: call spray_pdf_output
c
c
c ADDITIONAL I/O:
c
c    INPUTS: None.
c
c    OUTPUTS:
c
c    liquid_results_new
c    liquid_results_ini
c    spray_pdf_parameter_input
c
c --------------------------------------------------
c
c --------------------------------------------------
c ---
c
c Write spray restart files.
c
      if(lspray) then
        open(unit=idwrit,file='liquid_results_new',
     >  access='direct',recl=136,
     >  form='unformatted')
       if(ipid.eq.1) then
        open(unit=idwrit2,file='liquid_results_ini')
        write(idwrit2,*)nr_total
        call flush(idwrit2)
        write(idwrit2,*)dtil,dtml,t1,tl1,tm1
        call flush(idwrit2)
        write(idwrit2,*)iseed
        call flush(idwrit2)
        close(unit=idwrit2)
       endif
c
       INTS_DATA_2=314
       do n=1,np
       no_to_ip(n)=nr
       enddo
       do n=1,np
       if(ipid.ne.n) then
       irc= send_data_i (iul(n),no_to_ip(n), 1, INTS_DATA_2)
       endif
       enddo
       do n=1,np
       if(ipid.ne.n) then
       irc= recv_data_i (iul(n),no_fr_ip(n),  1, INTS_DATA_2)
       endif
       enddo
       no_fr_ip(ipid)=no_to_ip(ipid)
       irecordd=0
       do n=1,ipid-1
       irecordd=irecordd+no_fr_ip(n)
       enddo
c
       do ip=1,nr
        irecord=irecordd+ip
        isent=isen(ip)+(isep(ip)-1)*nodes
        write(idwrit,rec=irecord) ndrr(ip),ins(ip),
     1  isent,xki(ip),yki(ip),zki(ip),uki(ip),
     2  vki(ip),wki(ip),tki(ip),rki(ip),ski(ip),sklim(ip),
     3  (vh(ip,j),j=1,nde+4)
        call flush(idwrit)
       if(ip.ge.1) then
       write(1,*) irecord,ndrr(ip),ins(ip),
     1 isent,xki(ip),yki(ip),zki(ip),uki(ip),
     2 vki(ip),wki(ip),tki(ip),rki(ip),ski(ip),sklim(ip),
     3 (vh(ip,j),j=1,nde+4),nr,nr_total,irecord
       endif
       enddo
        close(unit=idwrit)
      endif
c
c -------------------------------------------------
c
c -------------------------------------------------
c ---
c
c Update file: spray_pdf_parameter_input.
c Also, write PDF restart files.
```

```
c
c
      if(ipdf.eq.1) then
        if(ipid.eq.1) then
        open(unit=85,file='spray_pdf_parameter_input')
        write(85,*)'lspray ldread ispray_mod'
        write(85,*)lspray,ldread,ispray_mod
        write(85,*)'ipread idread idwrit idread2 idwrit2'
        write(85,*)ipread,idread,idwrit,idread2,idwrit2
        write(85,*)'ipdf ns ipdf_mod ipdf_num'
        write(85,*)ipdf,ns,ipdf_mod,ipdf_num
        write(85,*)'irea1 irea2 iwri1 iwri2'
        write(85,*)irea1,irea2,iwri1,iwri2
        close(unit=85)
        endif
        call outpdf2(ns)
      endif
c
c --------------------------------------------------
c
c --------------------------------------------------
c ---
c
c Write output of spray computations either to unit
c one or to the screen.
c
      if(lspray) call prnspr
c
      return
      END
c
      subroutine outpdf2(ncyc)
      include 'p3dpar.i'

      include 'cfsparms.i'
      include 'cfsnodes.i'

      include 'p3dcom.i'
      include 'p3dave.i'

       include 'parallel.i'
       include 'cfsmimd.i'
       include 'dcfslog_rw.i'

c ----------------------------------------------------------
c ---
c
c PURPOSE: Outputs the PDF solution to restart files.
c
c FORM OF CALL: call outpdf2()
c
c ARGUMENTS/PARAMETERS: ncyc
c
c ADDITIONAL I/O: pdf_results, pdf_results_ave
c
c OUTPUT: outputs PDF solution to pdf_results &
c         outputs PDF averaging-procedure solution
c         to pdf_results_ave.
c ---
c ----------------------------------------------------------
      itrecl=nparti*nscala*4+4
      open(unit=iwri1,file='pdf_results',
     >access='direct',recl=itrecl,form='unformatted')
c
      itrecl=nscala*nav*4+nscala*2*4+4
      open(unit=iwri2,file='pdf_results_ave',
     >access='direct',recl=itrecl,form='unformatted')
c
      do ijk=1,nodes
      irecord=ijk+nodes*(ipid-1)
      write(iwri1,rec=irecord)
     >((part2(ijk,i1,i2),i1=1,nparti),i2=1,nscala)
      call flush(iwri1)
      enddo
c
      do ijk=1,nodes
      irecord=ijk+nodes*(ipid-1)
      write(iwri2,rec=irecord)
     >((av1(ijk,i1,i2),i1=1,nscala),i2=1,nav),
     >((avmany(ijk,i3,i4),i3=1,nscala),i4=1,2)
      call flush(iwri2)
      enddo
c
```

```
      close(unit=iwri1)
      close(unit=iwri2)
c
      return
      end
c
c
```

Appendix VIII

An Example of the Partial Listings of Code Initiation For Coupling LSPRAY and EUPDF With a Gas Flow Solver

1. The following segment shows how include calls to spray_int_rerun & pdf_int_rerun.

```
c
c -------------------------------------------------
c ---
c
c Include common blocks associated with spray and PDF computations.
c
      include 'dcfslog.i'
      include 'dcfslog_rw.i'
c
c
c Initialize Monte Carlo PDF computations.
c
      if(ipdf.eq.1) then
      call pdf_int_rerun
      endif
c
c Initialize spray computations.
c
      IF(lspray) then
      call spray_int_rerun
      endif
c
c -------------------------------------------------
```

II. The following segment shows how to include calls to DCLR & PDF.

```
c
c Include common blocks associated with spray and PDF computations.
c
      include 'dcfslog.i'
c
      double precision tbiggas, tendgas, totaltgas
```

```
c ---------------------------------------------------------
c ---
c
c
c Call dclr in order to advance the spray computations
c over a time step of dtgl.
c
      if(lspray.and.mod(iteration,ispray_mod).eq.0) then
       call dclr
      endif
c
c ---------------------------------------------------------
c
c ---------------------------------------------------------
c ---
c
c
c Call pdf in order to advance the PDF computations
c over the next time step.
c

      if(ipdf.eq.1.and.mod(iteration,ipdf_mod).eq.0) then
       do i=1,ipdf_num
        call pdf
       enddo
      endif
c
c ---------------------------------------------------------
c
```

III. The following segment shows how to include the interphase contributions to the gas phase governing equations.

```
c --------------------------------------------------
c ---
c
c Include common blocks associated with spray and PDF computations.
c
      include 'dcfslog.i'
c
c Include common blocks associated with spray the solver.
c
```

```
      include 'd3dqat.i'
c
c --------------------------------------------------
c
c --------------------------------------------------
c ---
c
c Include liquid-phase contributions to mass, momentum, species,
c and energy equations.
c
      if(lspray) then
        do i=1,nodes
          sourcem(i)=sourcem(i)+qmsx(i)
          sourceu(i)=sourceu(i)+qmsx(i)
          sourcev(i)=sourcev(i)+qmsy(i)
          sourcew(i)=sourcew(i)+qmsz(i)
          sourcef(i)=sourcef(i)+qms(i)
          sourceh(i)=sourceh(i)+qmsh(i)
        enddo
      endif
c
c --------------------------------------------------
```

Appendix IX

A Listing of Subroutines Used For Thermodynamic and Transport Property Evaluation

```
c
c
      subroutine rctinp

      include 'p3dpar.i'

      include 'cfsparms.i'
      include 'cfsnodes.i'

      include 'p3dcom.i'
      include 'p3dpro.i'
c      include 'p3dtim.i'

      dimension cpt0(nspt,3),cpt1(nspt,3),cpt2(nspt,3),
     >          cpt3(nspt,3),cpt4(nspt,3)
c
c ---------------------------------------------------------
c ---
c PURPOSE: Initializes parameters used in the themodynamic
c          and transport property evaluation as well as in
c          the chemical kinetics scheme.
c ---------------------------------------------------------
c
c ru = universal gas constant
c pz = pressure
c
      ru    =8314.3
      pz    =1.01325e+5
      cppdf =1297.1012
      trefpdf=298.0
c
c      default species:
c
c         o2       = 1
c         c7h16    = 2
c         h2o      = 3
c         co2      = 4
c         n2       = 5
```

```
c
c iconpc = store differences in array index convention used by the
c          flow and pdf codes.
c
      iconpc(1)=2
      iconpc(2)=1
      iconpc(3)=3
c
c read datafile on properties for n-heptane.
c
c w1,w2,w3     = constants resulting from some algebraic manipulation
c                which reduces the need for solving speciess equations
c                from 5 to 3.
c
c xnup, xnupp = stoichiometric coefficients of reactants and
c                products.
c
c arh,erh      = constants in the equation used for representing the
c                chemical reaction rate.
c
c wmole        = molecular weight
c
c cp0,..cp4    = coefficients of the polynomial used in determining the
c                variable specific heat.
c
c dwm          = coefficients of the polynomial used in determining the
c                inverse of the molecular weight of a mixture.
c
c dhform       = coefficients of the polynomial used in determining the
c                specific enthalpy.
c
c eno2, epox,
c epfu,emno2,
c empox,empfu = constants used in the chemical
c                kinetics solver.
c

      open(unit=35,file='nheptane_property_file')
      read(35,*)w1,w2,w3
      read(35,*)(xnup(1,i),i=1,4)
      read(35,*)(xnupp(1,i),i=1,4)
      read(35,*)arh(1),erh(1)
      read(35,*)(wmole(i),i=1,5)
c
      read(35,*)((cp0(i,j),i=1,5),j=1,3),
     1         ((cp1(i,j),i=1,5),j=1,3),
     1         ((cp2(i,j),i=1,5),j=1,3),
     1         ((cp3(i,j),i=1,5),j=1,3),
     1         ((cp4(i,j),i=1,5),j=1,3)
c
      read(35,*)(dwm(i),i=1,3)
c
      read(35,*)(dhform(i),i=1,3)
c
      read(35,*)eno2,epox,epfu,emno2,empox,empfu
      close(unit=35)
c
c
c
      return
      end


      subroutine props_ther
      include 'cfsparms.i'
      include 'cfsnodes.i'
      include 'cfsvars.i'
      include 'cfsh.i'
      include 'cfsprop.i'
      include 'cfspref.i'
      include 'cfsgas.i'
      include 'cfscpmix.i'
      include 'conjugate.i'
c
      include 'p3dpar.i'
      include 'p3dpro.i'
c -------------------------------------------------------------
c ---
c PURPOSE: This routine computes enthalpy, specific heat, and
c          density.
c -------------------------------------------------------------
c
      do 100 ijk=1,nodes
      if(fluid(ijk)) then
      tmp= t(ijk)
      icspec=iconpc(3)
      d0=dhform(icspec)
      icspec=iconpc(1)
      d0=d0+ f1(ijk)*dhform(icspec)
      icspec=iconpc(2)
      d0=d0+ f2(ijk)*dhform(icspec)
      do 1010 iter=1,1
      jc=cvmgp(2,1,tmp-600.0)
      jc=cvmgp(3,jc,tmp-1000.0)
      icspec=iconpc(3)
      d1= cp0(icspec,jc)
      d2= cp1(icspec,jc)
      d3= cp2(icspec,jc)
      d4= cp3(icspec,jc)
      d5= cp4(icspec,jc)
      icspec=iconpc(1)
      d1=d1+ f1(ijk)*cp0(icspec,jc)
      d2=d2+ f1(ijk)*cp1(icspec,jc)
      d3=d3+ f1(ijk)*cp2(icspec,jc)
      d4=d4+ f1(ijk)*cp3(icspec,jc)
      d5=d5+ f1(ijk)*cp4(icspec,jc)
      icspec=iconpc(2)
      d1=d1+ f2(ijk)*cp0(icspec,jc)
      d2=d2+ f2(ijk)*cp1(icspec,jc)
      d3=d3+ f2(ijk)*cp2(icspec,jc)
      d4=d4+ f2(ijk)*cp3(icspec,jc)
      d5=d5+ f2(ijk)*cp4(icspec,jc)
      h(ijk)=((((0.20*d5*tmp+0.25*d4)*tmp+d3/3.0)*tmp+
     >        0.50*d2)*tmp+d1)*tmp+d0
      devm=(((d5*tmp+d4)*tmp+
     >        d3)*tmp+d2)*tmp+d1
 1010 continue
      icspec=iconpc(3)
      rmw = dwm(icspec)
      icspec=iconpc(1)
      rmw = rmw + f1(ijk)*dwm(icspec)
      icspec=iconpc(2)
      rmw = rmw + f2(ijk)*dwm(icspec)
      rho(ijk)=(p(ijk)+p_reference)/(Rgas*tmp*rmw)
      cp(ijk) = devm
      endif
 100  continue
      return
      end



      subroutine props_tran
c
      include 'cfsparms.i'
      include 'cfsnodes.i'
      include 'cfsvars.i'
      include 'cfsmimd.i'
      include 'cfsprop.i'
      include 'cfsother.i'
      include 'cfsturb.i'
      include 'cfsvislam.i'
      include 'cfsdt.i'
      include 'cfscpmix.i'
      include 'conjugate.i'
c ----------------------------------------------------------
c ---
c PURPOSE: This routine computes transport properties.
c ----------------------------------------------------------
c
c --- calculate the effective viscosity for turbulent flow
         if(turbulent) then
      lvis=0
      do 100 ijk=1,nodes
      if(fluid(ijk)) then
      mu(ijk)=rho(ijk)*k(ijk)**2*cmu/(e(ijk)+1.e-20)
     &       +temp_vislam
c --- enforce upper and lower bounds on turbulent viscosity
      if(mu(ijk).gt.vismax) then
      lvis=1
      mu(ijk)=vismax
      endif
      mu(ijk)=amax1(mu(ijk),temp_vislam)
      kt(ijk)=mu(ijk)*cp(ijk)/prandl
      endif
 100  continue
      if(mod(itime,50).eq.0.and.ipid.eq.1) then
      if(lvis.eq.0) print 746
 746  format(t11,'No limit enforced on turbulent viscosity.',/)
      if(lvis.eq.1) print 747
 747  format(t11,'Limit enforced on turbulent viscosity.',/)
      endif
         else
      do 101 ijk=1,nodes
      if(fluid(ijk)) then
      mu(ijk)=temp_vislam
      kt(ijk)=mu(ijk)*cp(ijk)/prandl
      endif
 101  continue
         endif
      return
      end

      subroutine get_kt_and_cp_loc_mcs(ijk,centroid)
      include 'cfsparms.i'
      include 'cfsnodes.i'
      include 'cfsvars.i'
      include 'cfsprop.i'
      include 'cfsarea.i'
      include 'cfsflux.i'
      include 'cfssorc_enth.i'
      include 'cfschar.i'
      include 'cfsadj.i'
      include 'cfsdt.i'
      include 'cfsorder.i'
      include 'cfscomp.i'
      include 'cfscpmix.i'
      include 'conjugate.i'
c
      include 'p3dpar.i'
      include 'p3dpro.i'

      logical centroid
c -----------------------------------------------------------
c ---
c PURPOSE: Computes specific heat and thermal conductivity of a
c          gaseous mixture at a nodal point of the computational
c          grid.
c -----------------------------------------------------------
c
```

```
 if(centroid) then
 tmp= t(ijk)
 jc=cvmgp(2,1,tmp-600.0)
 jc=cvmgp(3,jc,tmp-1000.0)
 icspec=iconpc(3)
 d1= cp0(icspec,jc)
 d2= cp1(icspec,jc)
 d3= cp2(icspec,jc)
 d4= cp3(icspec,jc)
 d5= cp4(icspec,jc)
 icspec=iconpc(1)
 d1=d1+ f1(ijk)*cp0(icspec,jc)
 d2=d2+ f1(ijk)*cp1(icspec,jc)
 d3=d3+ f1(ijk)*cp2(icspec,jc)
 d4=d4+ f1(ijk)*cp3(icspec,jc)
 d5=d5+ f1(ijk)*cp4(icspec,jc)
 icspec=iconpc(2)
 d1=d1+ f2(ijk)*cp0(icspec,jc)
 d2=d2+ f2(ijk)*cp1(icspec,jc)
 d3=d3+ f2(ijk)*cp2(icspec,jc)
 d4=d4+ f2(ijk)*cp3(icspec,jc)
 d5=d5+ f2(ijk)*cp4(icspec,jc)
 cp(ijk)=(((d5*tmp+d4)*tmp+d3)*tmp+
>             d2)*tmp+d1
 kt(ijk)=mu(ijk)*cp(ijk)/prandl
 else
 tmp= tbar(ijk)
 jc=cvmgp(2,1,tmp-600.0)
 jc=cvmgp(3,jc,tmp-1000.0)
 icspec=iconpc(3)
 d1= cp0(icspec,jc)
 d2= cp1(icspec,jc)
 d3= cp2(icspec,jc)
 d4= cp3(icspec,jc)
 d5= cp4(icspec,jc)
 icspec=iconpc(1)
 d1=d1+ f1bar(ijk)*cp0(icspec,jc)
 d2=d2+ f1bar(ijk)*cp1(icspec,jc)
 d3=d3+ f1bar(ijk)*cp2(icspec,jc)
 d4=d4+ f1bar(ijk)*cp3(icspec,jc)
 d5=d5+ f1bar(ijk)*cp4(icspec,jc)
 icspec=iconpc(2)
```

```
      d1=d1+ f2bar(ijk)*cp0(icspec,jc)
      d2=d2+ f2bar(ijk)*cp1(icspec,jc)
      d3=d3+ f2bar(ijk)*cp2(icspec,jc)
      d4=d4+ f2bar(ijk)*cp3(icspec,jc)
      d5=d5+ f2bar(ijk)*cp4(icspec,jc)
      cpbar(ijk)=(((d5*tmp+d4)*tmp+d3)*tmp+
     >             d2)*tmp+d1
      ktbar(ijk)=mubar(ijk)*cpbar(ijk)/prandl
      endif
      RETURN
      END

      function find_inverse_molecular_weight_mcs(element,centroid)
      include 'cfsparms.i'
      include 'cfsvars.i'
      include 'cfsgas.i'
c      include 'cfschem.i'
      integer element
      logical centroid
c
      include 'p3dpar.i'
      include 'p3dpro.i'
c ---------------------------------------------------------
c ---
c PURPOSE: Computes inverse of the molecular weight of a gaseous
c          mixture at a nodal point of the computational grid.
c ---------------------------------------------------------
c
         icspec=iconpc(3)
         find_inverse_molecular_weight_mcs= dwm(icspec)
       if(centroid) then
         icspec=iconpc(1)
         find_inverse_molecular_weight_mcs=
     >   find_inverse_molecular_weight_mcs+f1(element)*dwm(icspec)
         icspec=iconpc(2)
         find_inverse_molecular_weight_mcs=
     >   find_inverse_molecular_weight_mcs+f2(element)*dwm(icspec)
       else
         icspec=iconpc(1)
         find_inverse_molecular_weight_mcs=
     >   find_inverse_molecular_weight_mcs+f1bar(element)*dwm(icspec)
         icspec=iconpc(2)
```

```
           find_inverse_molecular_weight_mcs=
     >     find_inverse_molecular_weight_mcs+f2bar(element)*dwm(icspec)
         endif
C
        return
        end

        function find_h_mcs(element,centroid)
        include 'cfsparms.i'
        include 'cfsvars.i'
        include 'cfshref.i'
        include 'cfsgas.i'
        include 'cfscpmix.i'
c        include 'cfschem.i'
c        include 'BLOCK'
        integer element
        logical centroid
c
        include 'p3dpar.i'
        include 'p3dpro.i'
c ---------------------------------------------------------
c ---
c PURPOSE: Computes specific enthalpy at a nodal point of the
c          computational grid.
c ---------------------------------------------------------
c
           if(centroid) then
        tmp = t(element)
        jc=cvmgp(2,1,tmp-600.0)
        jc=cvmgp(3,jc,tmp-1000.0)
        icspec=iconpc(3)
        hfmm0=dhform(icspec)
        d1= cp0(icspec,jc)
        d2= cp1(icspec,jc)
        d3= cp2(icspec,jc)
        d4= cp3(icspec,jc)
        d5= cp4(icspec,jc)
        icspec=iconpc(1)
        hfmm0=hfmm0+ f1(element)*dhform(icspec)
        d1=d1+ f1(element)*cp0(icspec,jc)
        d2=d2+ f1(element)*cp1(icspec,jc)
        d3=d3+ f1(element)*cp2(icspec,jc)
```

```
      d4=d4+ f1(element)*cp3(icspec,jc)
      d5=d5+ f1(element)*cp4(icspec,jc)
      icspec=iconpc(2)
      hfmm0=hfmm0+ f2(element)*dhform(icspec)
      d1=d1+ f2(element)*cp0(icspec,jc)
      d2=d2+ f2(element)*cp1(icspec,jc)
      d3=d3+ f2(element)*cp2(icspec,jc)
      d4=d4+ f2(element)*cp3(icspec,jc)
      d5=d5+ f2(element)*cp4(icspec,jc)
c --- calculate enthalpy of mixture
      find_h_mcs=((((0.20*d5*tmp+0.25*d4)*tmp+d3/3.0)*tmp+
     >           0.50*d2)*tmp+d1)*tmp+hfmm0
         else
      tmp = tbar(element)
      jc=cvmgp(2,1,tmp-600.0)
      jc=cvmgp(3,jc,tmp-1000.0)
      icspec=iconpc(3)
      hfmm0=dhform(icspec)
      d1= cp0(icspec,jc)
      d2= cp1(icspec,jc)
      d3= cp2(icspec,jc)
      d4= cp3(icspec,jc)
      d5= cp4(icspec,jc)
      icspec=iconpc(1)
      hfmm0=hfmm0+ f1bar(element)*dhform(icspec)
      d1=d1+ f1bar(element)*cp0(icspec,jc)
      d2=d2+ f1bar(element)*cp1(icspec,jc)
      d3=d3+ f1bar(element)*cp2(icspec,jc)
      d4=d4+ f1bar(element)*cp3(icspec,jc)
      d5=d5+ f1bar(element)*cp4(icspec,jc)
      icspec=iconpc(2)
      hfmm0=hfmm0+ f2bar(element)*dhform(icspec)
      d1=d1+ f2bar(element)*cp0(icspec,jc)
      d2=d2+ f2bar(element)*cp1(icspec,jc)
      d3=d3+ f2bar(element)*cp2(icspec,jc)
      d4=d4+ f2bar(element)*cp3(icspec,jc)
      d5=d5+ f2bar(element)*cp4(icspec,jc)
c --- calculate enthalpy of mixture
      find_h_mcs=((((0.20*d5*tmp+0.25*d4)*tmp+d3/3.0)*tmp+
     >           0.50*d2)*tmp+d1)*tmp+hfmm0
         endif
c
```

```
      return
      end
c
```

An Example Summary of CPU Times Taken By CORSAIR and EUPDF

Table 1 summarizes the cpu times per cycle taken by EUPDF and CORSAIR solvers versus the number of processors used on the NASA LeRC LACE cluster. These computations refer to the case of a confined swirl-stabilized spray flame as reported in Ref. 7. The computations were performed on a grid of 3600 quadrilateral elements and a total of 0.36 million Monte Carlo particles (=100 particles/cell). Both the PDF and CFD solvers showed excellent parallel performance with an increase in the number of processors. For a discussion of the parallel performance of these solvers refer to Ref. 7. It takes approximately about 1000 to 2000 cycles for the computations to reach a converged solution.

Table 1. Cpu time (sec) per cycle versus number of PEs on LACE Cluster.

		Number of processors			
Solver	Characteristic	2	4	8	16
EUPDF	1 step/cycle	2.30	1.35	0.75	0.44
CORSAIR	5 steps/cycle	3.55	1.90	1.10	0.60

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE April 1998	3. REPORT TYPE AND DATES COVERED Final Contractor Report

4. TITLE AND SUBTITLE EUPDF - An Eulerian-Based Monte Carlo Probability Density Function (PDF) Solver User's Manual	5. FUNDING NUMBERS WU-523-26-33-00 NAS3-98022
6. AUTHOR(S) M.S. Raju	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NYMA, Inc. 2001 Aerospace Parkway Brook Park, Ohio 44142	8. PERFORMING ORGANIZATION REPORT NUMBER E-11148
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA CR—1998-207401

11. SUPPLEMENTARY NOTES

Project Manager, N.S. Liu, Turbomachinery and Propulsion Systems Division, NASA Lewis Research Center, organization code 5830, (216) 433-8722.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Categories: 07, 20, 34, 61, and 64 Distribution: Nonstandard This publication is available from the NASA Center for AeroSpace Information, (301) 621-0390.	12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

EUPDF is an Eulerian-based Monte Carlo PDF solver developed for application with sprays, combustion, parallel computing and unstructured grids. It is designed to be massively parallel and could easily be coupled with any existing gas-phase flow and spray solvers. The solver accommodates the use of an unstructured mesh with mixed elements of either triangular, quadrilateral, and/or tetrahedral type. The manual provides the user with the coding required to couple the PDF code to any given flow code and a basic understanding of the EUPDF code structure as well as the models involved in the PDF formulation. The source code of EUPDF will be available with the release of the National Combustion Code (NCC) as a complete package.

14. SUBJECT TERMS Computational combustion; Spray combustion; Monte Carlo PDF methods; Combustion	15. NUMBER OF PAGES 71
	16. PRICE CODE A04

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT